Exploiting the Trade-Off — The Benefits of Multiple Objectives in Data Clustering

Julia Handl and Joshua Knowles

School of Chemistry, University of Manchester
Faraday Building, Sackville Street, PO Box 88, Manchester M60 1QD
`http://dbk.ch.umist.ac.uk/handl/mock/`

Abstract. In previous work, we have proposed a novel approach to data clustering based on the explicit optimization of a partitioning with respect to two complementary clustering objectives [6]. Here, we extend this idea by describing an advanced multiobjective clustering algorithm, MOCK, with the capacity to identify good solutions from the Pareto front, and to automatically determine the number of clusters in a data set. The algorithm has been subject to a thorough comparison with alternative clustering techniques and we briefly summarize these results. We then present investigations into the mechanisms at the heart of MOCK: we discuss a simple example demonstrating the synergistic effects at work in multiobjective clustering, which explain its superiority to single-objective clustering techniques, and we analyse how MOCK's Pareto fronts compare to the performance curves obtained by single-objective algorithms run with a range of different numbers of clusters specified.

Keywords: Clustering, multiobjective optimization, evolutionary algorithms, automatic determination of the number of clusters.

1   Introduction

Clustering is commonly defined as the task of finding natural groups within a data set such that data items within the same group are more similar than those within different groups. This is an intuitive but rather 'loose' concept, and it remains quite difficult to realize in general practice. Evidently, one reason for the difficulty is that, for many data sets, no unambiguous partitioning of the data exists, or can be established, even by humans. But even in cases where an unambiguous partitioning of the data *is* possible, clustering algorithms can drastically fail. This is because most existing clustering techniques rely on estimating the quality of a particular partitioning by means of just one *internal evaluation function*, an objective function that measures intrinsic properties of a partitioning, such as the spatial separation between clusters or the compactness of clusters. Hence, the internal evaluation function is assumed to reflect the quality of the partitioning reliably, an assumption that may be violated for certain data sets.

Given that many objective functions for clustering are complementary, the simultaneous optimization of several such objectives may help to overcome this weakness. In previous work [6], we have demonstrated this idea, showing that the simultaneous optimization of two clustering objectives results in clear performance gains with respect to single-objective clustering algorithms. However, the algorithm presented, VIENNA (Voronoi initialized evolutionary nearest neighbour algorithm), was limited in two respects: its application required knowledge of the correct number of clusters, and no mechanism was presented to select good solutions from the Pareto front obtained.

Our new algorithm, MOCK (multiobjective clustering with automatic determination of the number of clusters), overcomes these weaknesses. It uses a novel, flexible representation that permits us to efficiently generate clustering solutions that both correspond to different trade-offs between our two clustering objectives and that contain different numbers of clusters. An automated technique is employed to select high-quality solutions from the resulting Pareto front, and it thus simultaneously determines the number of clusters in a data set. We briefly present the algorithm in this paper and summarize analytical results demonstrating its robust performance across data sets that exhibit a wide range of different data properties. A more detailed description and additional analytical results are provided in [7]. Besides introducing MOCK, a second goal of this paper is to give more insight into the mechanisms underlying multiobjective clustering: in particular, we aim to show that MOCK's good performance arises as a direct consequence of the simultaneous optimization of several clustering objectives: an archetypal problem — serving to illustrate the synergistic effects at work in multiobjective clustering — is used for this purpose. We additionally underline the differences between single- and multiobjective clustering by comparing the shape of MOCK's Pareto fronts to the performance curves obtained for single-objective clustering methods run with a varying number of clusters specified.

The remainder of this paper is organized as follows. Section 2 briefly summarizes related work on clustering and evolutionary algorithms. This is followed by a description of our algorithm, MOCK (Section 2.1), and all other contestant methods used in this study (Section 3). Section 4 presents our experiments and discusses results, and Section 5 concludes.

2   Related Work

Clustering problems arise in a variety of different disciplines, ranging from biology to sociology to computer science. Consequently, they have been the subject of active research for several decades, and a multitude of clustering methods exist nowadays, which fundamentally differ in their basic principles and in the properties of the data they can tackle. For an extensive survey of clustering problems and algorithms the reader is referred to Jain et al. [8].

Evolutionary algorithms (EAs) have a history of being applied to clustering problems [4, 12, 13]. However, previous research in this respect has been lim-

ited to the single objective case: k-means' criterion of intra-cluster variance has been the objective most commonly employed, as this measure provides smooth incremental guidance in all parts of the search space. Due to the difficulty of deriving an encoding and operators that efficiently explore the very large clustering search space, actual hybridizations between EAs and k-means have also been particularly popular [12, 13].

Multiobjective evolutionary algorithms (MOEAs) have repeatedly been used to perform feature selection for clustering [5, 9], but have not previously been applied to the actual clustering task itself. This is despite the general agreement that clustering objectives can be conflicting or complementary, and that no single clustering objective can deal with every conceivable cluster structure [10]. To date, attempts to deal with this problem have focused on the *retrospective* combination of different clustering results by means of ensemble methods [11, 16, 18]. In order to construct clustering ensembles, different clustering results are first generated by repeatedly running the same algorithm (using different initializations, bootstrapping or a varying number of clusters) or several complementary methods (e.g. agglomerative algorithms based on diverse linkage criteria such as single-link and average-link). The resulting solutions are then combined into an ensemble clustering using graph-based approaches, expectation maximization or co-association methods.

Results reported in the literature demonstrate that clustering ensembles are often more robust and yield higher quality results than individual clustering methods, indicating that the combination of several clustering objectives is favourable. However, it is our contention that ensemble methods do not fully exploit the potential of using several objectives, as they are limited to the *a posteriori* integration of solutions rather than exploring trade-off solutions *during* the clustering process. We aim to overcome this limitation by tackling clustering as a truly multiobjective optimization problem.

2.1   Multiobjective Clustering

We based our multiobjective clustering algorithm on the elitist MOEA, PESA-II, described in detail in [3].

PESA-II. Briefly, PESA-II updates, at each generation, a current set of nondominated solutions stored in an external population (of non-fixed but limited size), and uses this to build an internal population of fixed size to undergo reproduction and variation. PESA-II uses a selection policy designed to give equal reproduction opportunities to all regions of the current nondominated front; thus in the clustering application, it should provide a diverse set of solutions trading off different clustering measures. No critical parameters are associated with this 'niched' selection policy, as it uses an adaptive range equalization and normalization of the objectives. PESA-II may be used to optimize any number of objective functions, allowing us to simultaneously optimize several clustering measures, but in our algorithm MOCK we use just two (conceptually distant) measures as objectives, described below.
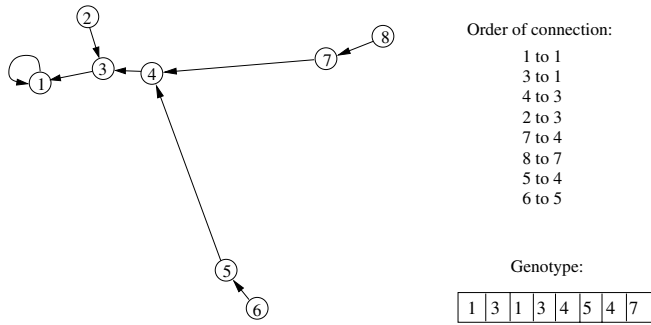
Fig. 1. Construction of the minimum spanning tree and its genotype coding. The data item with label 1 is first connected to itself, then Prim's algorithm is used to connect the other items. In the genotype, each gene (i.e. position in the string) represents the respective data item, and its allele value represents the item it points to (e.g. gene 2 has allele value 3 because data item 2 points to data item 3). The genotype coding for the full MST (as shown) is used as the first individual in the EA population

Genetic Representation and Operators. To apply PESA-II to the clustering problem, a suitable genetic encoding of a partitioning and one or more genetic variation operators (e.g. mutation and/or crossover) have to be chosen.

We employ the locus-based adjacency representation proposed in [14]. In this graph-based encoding (see Figure 1), each individual g consists of N genes $g_1, \ldots, g_N$, where N is the size of the data set given, and each gene g_i can take allele values j in the range $\{1, \ldots, N\}$. Thus, a value of j assigned to the ith gene, is then interpreted as a link between data items i and j: in the resulting clustering solution they will be in the same cluster. The decoding of this representation requires the identification of all subgraphs, which can be done in linear time. All data items belonging to the same subgraph are then assigned to one cluster.

This encoding scheme permits us to keep the number of clusters dynamic and is well-suited for use with standard crossover operators such as uniform, one-point or two-point crossover. We choose uniform crossover, and employ a specialized mutation operator that significantly reduces the size of the search space: each data item can only be linked to one of its L nearest neighbours. Hence, $g_i \in \{nn_{i1}, \ldots, nn_{iL}\}$, where nn_{il}, $l \in 1 \ldots L$, denotes the lth nearest neighbour of data item i.

Our initialization routine also exploits the link-based encoding and uses minimum spanning trees (MSTs). For a given data set, we first compute the complete MST using Prim's algorithm. The ith individual of the initial populations is then initialized by the MST with the $(i-1)$th largest links removed (see Figure 1).

Objective Functions. MOCK's clustering objectives have been chosen to reflect two fundamentally different aspects of a good clustering solution: the global concept of *compactness of clusters*, and the more local one of *connectedness of data points*.

In order to express cluster compactness we calculate the *overall deviation* of a partitioning. This is simply computed as the overall summed distances between data items and their corresponding cluster centre:

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k),$$

where C is the set of all clusters, μ_k is the centre of cluster C_k and $\delta(.,.)$ is the distance function chosen (Euclidean distance in this paper). As an objective, overall deviation should be minimized.

As an objective reflecting cluster connectedness, we use a measure, connectivity, which evaluates the degree to which neighbouring data-points have been placed in the same cluster. It is computed as

$$Conn(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), \quad x_{i,nn_i(j)} = \begin{cases} \frac{1}{j} & \text{if } \nexists C_k : i, nn_i(j) \in C_k \\ 0 & \text{otherwise,} \end{cases}$$

where $nn_i(j)$ is the jth nearest neighbour of datum i, and L is a parameter determining the number of neighbours that contribute to the connectivity measure. As an objective, connectivity should be minimized.

2.2   Automatic Solution Selection

The MOEA described above returns a set of nondominated solutions corresponding to different compromises of the two objectives, and containing different numbers of clusters. We next devise an automated method to select a single one of these solutions as the correct 'answer'. Unlike previous approaches to identifying promising solutions from the Pareto front (e.g. [2]), our method is application-specific. In particular, it makes use of several domain-specific considerations (inspired by Tibshirani et al.'s Gap statistic [17]) that enable us to derive a more effective technique for our particular purpose.

Intuitively, we expect the structure of the data to be reflected in the shape of the Pareto front. From the two objectives employed, overall deviation decreases with an increasing number of clusters, whereas connectivity increases. Hence, we can say that, incrementing the number of clusters k, we gain an improvement in overall deviation δD at the cost of a degradation in connectivity δC. For a number of clusters k smaller than the true number, we expect the ratio $R = \frac{\delta D}{\delta C}$ to be large: the separation of two clusters will trigger a great decrease in overall deviation, with only a small or no increase in connectivity. When we surpass the correct number of clusters this ratio will diminish: the decrease in overall deviation will be less significant but come at a high cost in terms of connectivity (because a true cluster is being split). Due to the natural bias of both measures, the solutions in the Pareto front are approximately ordered by the number of clusters they contain: plotting connectivity on the abscissa and overall deviation on the ordinate, k gradually increases from left

to right. The distinct change in R occurring for the correct number of clusters can therefore be seen as a 'knee'. In order to help us correctly determine this knee, we use uniformly random reference data: clustering a number of such reference distributions using MOCK provides us with a set of 'reference fronts', which help us to abstract from k-specific biases in our clustering objectives (see Figure 2).
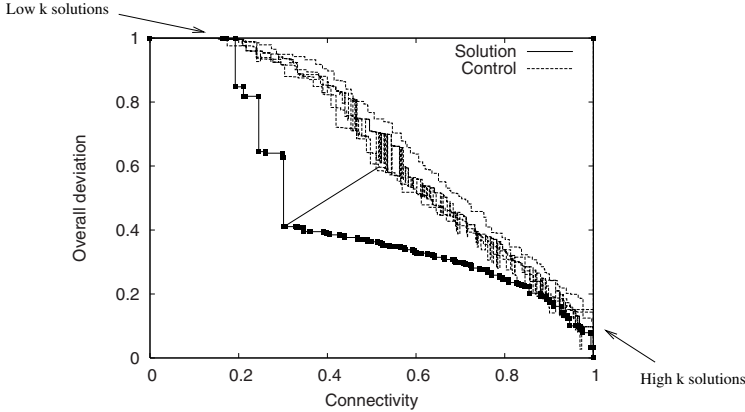

Fig. 2. Solution and control reference fronts for a run of MOCK on the *Square1* data set. The solution with the largest minimum distance to the reference fronts is indicated by the angled line, and corresponds to the correct $k = 4$ cluster solution

Briefly, we then determine good solutions as follows: after a normalization step, the distances between individual solutions and the attainment surfaces described by the reference fronts are computed. We plot the resulting *attainment scores* as a function of the number of clusters k. The maximum of the resulting curve provides us with the number of clusters k; also, the solution corresponding to the highest attainment score for this k is selected as the best solution. A more detailed motivation and description of this methodology (including pseudo-code) is provided in [7].

3   Contestant Methods

We evaluate MOCK by comparing it to four established single-objective clustering methods, whose implementations are described below. Three of these contestants are traditional and conceptually different clustering algorithms, namely k-means, single-link agglomerative clustering and average-link agglomerative clustering. Here, the algorithms k-means and single-link agglomerative clustering are of particular interest, as each of them uses a clustering objective that is conceptually quite similar to one of MOCK's. The fourth algorithm is an advanced clustering ensemble method by Strehl and Ghosh [16]. All four algorithms are

— differently to MOCK — given the same advantage of being provided with the correct number of clusters. A comparison of MOCK to a state-of-the-art method for the determination of the number of clusters, Tibshirani et al.'s Gap statistic [17], can be found in [7].

3.1 k-Means

Starting from a random partitioning, the k-means algorithm repeatedly (i) computes the current cluster centres (i.e. the average vector of each cluster in data space) and (ii) reassigns each data item to the cluster whose centre is closest to it. It terminates when no more reassignments take place. By this means, the intra-cluster variance, that is, the sum of squares of the differences between data items and their associated cluster centres, is locally minimized.

Our implementation of the k-means algorithm is based on the batch version of k-means, that is, cluster centres are only recomputed after the reassignment of all data items. As k-means can sometimes generate empty clusters, these are identified in each iteration and are randomly reinitialized. This enforcement of the correct number of clusters can prevent convergence, and we therefore set the maximum number of iterations to 100. To reduce suboptimal solutions k-means is run repeatedly (100 times) using random initialisation (which is known to be an effective initialization method [15]) and only the best result in terms of intra-cluster variance is returned.

3.2   Hierarchical Clustering

In general, agglomerative clustering algorithms start with the finest partitioning possible (i.e. singletons) and, in each iteration, merge the two least distant clusters. They terminate when the target number of clusters has been obtained. Single-link and average-link agglomerative clustering only differ in the linkage metric used. For the linkage metric of average-link, the distance between two clusters C_i and C_j is computed as the average dissimilarity between all possible pairs of data elements i and j with $i \in C_i$ and $j \in C_j$. For the linkage metric of single-link, the distance between two clusters C_i and C_j is computed as the smallest dissimilarity between all possible pairs of data elements i and j with $i \in C_i$ and $j \in C_j$.

3.3   Cluster Ensemble

Strehl and Ghosh's 'knowledge reuse framework' employs three conceptually different ensemble methods namely (1) CSPA (Cluster-based Similarity Partitioning Algorithm), (2) HGPA (Hyper-Graph Partitioning Algorithm) and (3) MCLA (Meta-Clustering Algorithm). The solutions returned by the individual combiners then serve as the input to a supra-consensus function, which selects the best solution in terms of average shared mutual information.

For the implementation of this cluster ensemble we use Strehl and Ghosh's original Matlab code with the correct number of clusters provided. In order to

generate the input labels we use the algorithms described above, that is, k-means, average-link and single-link agglomerative clustering. As ensemble methods generally benefit from being provided partitionings of higher resolution (i.e. comprising more clusters), we run each algorithm for all $k \in \{2, \ldots, 20\}$. The resulting 57 labelings then serve as the input to Strehl and Ghosh's method.

3.4   Parameter Settings for MOCK

Parameter settings for MOCK are given in Table 3.4 and are kept constant over all experiments.

Table 1. Parameter settings for MOCK, where N is data set size

Parameter	setting
Number of generations	200
External population size	1000
Internal population size	$\max(50, \frac{N}{20})$
Initialization	Minimum spanning tree
Mutation type	L nearest neighbours ($L = 20$)
Mutation rate p_m	$1/N$
Recombination	Uniform crossover
Recombination rate p_r	0.7
Objective functions	Overall deviation and connectivity ($L = 20$)
Constraints	$k \in \{1, \ldots, 25\}$, cluster size > 2
Number of reference distributions	5

4   Experiments

The following experimental section is split into two major parts. We first provide a summary of our comparative study between MOCK and the clustering methods introduced above. MOCK's high performance leads us to to investigate the reasons for the robustness of multiobjective clustering; an archetypal example and the visualization of solutions in two-objective space are used for this purpose.

4.1   Analytical Evaluation

In our comparative study, MOCK has been evaluated using a range of synthetic and real data sets. In this paper, we present results on 19 two-dimensional data sets that are well-suited to demonstrate MOCK's performance for different data properties. The particular properties studied are overlap between clusters (on the *Square* and *Triangle* series), unequally sized clusters (in terms of the number of data points contained; on the *Sizes* series) and elongated cluster shapes (on the *Long*, *Spiral* and *Smile* series). Detailed descriptions and generators for the data sets are available at [1]. More results, including those for high-dimensional and real data, can be found in [7].

The clustering results of the five different algorithms are compared using an objective evaluation function, the *F-Measure* [19], which is an external evaluation

Table 2. Sample Median and Interquartile Range F-Measure [19] values for 50 runs of each algorithm on two-dimensional synthetic data sets exhibiting different data properties. The best value is shown in bold. MOCK is the only consistent method; all other algorithms have a 'nemesis' data set on which they are worst

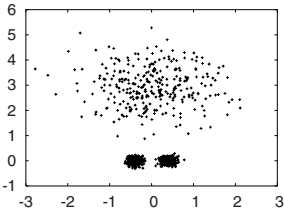| Problem	single-link	average-link	k-means	clustering ensemble	$MOCK$
Long1	0.666444 (0.333556)	0.665104 (0.005896)	0.521989 (0.015211)	**1.0** (0.0)	**1.0** (0.0)
Long2	0.678444 (0.000178)	0.67714 (0.011155)	0.520026 (0.01683)	0.902 (0.0)	**1.0** (0.0)
Long3	0.777895 (0.000556)	0.77514 (0.009774)	0.566661 (0.017321)	0.730591 (0.001802)	**1.0** (0.006)
Sizes1	0.428323 (0.000242)	0.977935 (0.007071)	**0.989003** (0.005013)	0.747616 (0.030904)	0.987 (0.006)
Sizes2	0.522742 (0.000477)	0.981947 (0.009885)	0.987051 (0.004999)	0.633283 (0.002187)	**0.988** (0.006)
Sizes3	0.600841 (0.000782)	0.98502 (0.00905)	0.987114 (0.006899)	0.562078 (0.00984)	**0.99** (0.005)
Sizes4	0.658308 (0.000676)	0.983953 (0.005826)	0.985274 (0.006851)	0.506595 (0.261149)	**0.989** (0.0041)
Sizes5	0.702411 (0.001261)	0.986976 (0.007064)	0.984288 (0.005843)	0.487591 (0.311809)	**0.9909** (0.0079)
Smile1	**1.0** (0.0)	0.753036 (0.0)	0.665609 (0.009407)	**1.0** (0.0)	**1.0** (0.0)
Smile2	**1.0** (0.0)	0.725156 (0.0)	0.586508 (0.009967)	0.91 (0.0)	**1.0** (0.0)
Smile3	**1.0** (0.0)	0.549761 (0.0)	0.505994 (0.007393)	0.776494 (0.001284)	**1.0** (0.0)
Spiral	**1.0** (0.0)	0.576 (0.0)	0.593 (0.002)	1.0 (0.0)	**1.0** (0.0)
Square1	0.399759 (8e-05)	0.977997 (0.015005)	**0.987006** (0.004982)	0.984 (0.008006)	0.985 (0.0051)
Square2	0.399759 (0.0)	0.961982 (0.009888)	**0.976019** (0.007988)	0.97 (0.008002)	0.973 (0.009)
Square3	0.399759 (8e-05)	0.934935 (0.016238)	**0.956933** (0.00802)	0.94599 (0.015982)	0.946 (0.0172)
Square4	0.399759 (8e-05)	0.883035 (0.02214)	**0.919999** (0.008024)	0.908 (0.019006)	0.9041 (0.0184)
Square5	0.399759 (8e-05)	0.720672 (0.107357)	**0.86798** (0.014231)	0.842965 (0.033088)	0.8361 (0.0324)
Triangle1	**1.0** (0.0)	0.997 (0.004001)	0.98486 (0.00613)	0.999 (0.001)	**1.0** (0.0)
Triangle2	0.45193 (0.116834)	0.986979 (0.013638)	0.957697 (0.011837)	0.810492 (0.068513)	**0.995** (0.004)

Fig. 3. A simple three-cluster data set posing difficulties to many clustering methods

function that requires knowledge of the correct class labels and compares a generated clustering solution to this 'gold standard'. The F-Measure can take on values in the interval $[0, 1]$ and should be maximized.

From the results presented in Table 2 it becomes clear that the single-objective algorithms all experience trouble for particular data properties. For k-means and average-link, the problematic data sets are those with arbitrary shaped or elongated clusters (in the *Long*, *Smile* and *Spiral* series) for single-link it is those that contain clusters with overlap or noise points (in the *Square*, *Sizes* and *Long* series). The clustering ensemble fares better, but severely breaks down for unequally sized clusters (in the *Sizes* series). Only MOCK shows a very strong performance for all types of different data properties and is best, or close to best, on almost all data sets.

4.2   A Simple Example Showing the Synergistic Effects at Work in MOCK

Given MOCK's performance, it is worth examining more closely the mechanisms underlying multiobjective clustering. In particular, we would like to show that MOCK's *access* to two conceptually different clustering objectives is *not sufficient* to explain its good performance, but that it is their *simultaneous optimization* that is the key to its success.

We first observe that, in an explicit single-objective optimization, it wouldn't be possible to keep the number of clusters dynamic (without the introduction of additional constraints) at all: due to the natural bias of both measures, an optimization method would, for *any data set*, necessarily converge to trivial solutions: these are singleton clusters for overall deviation and a single cluster for connectivity. Only the simultaneous optimization allows us to effectively explore interesting solutions, as we can exploit the oppositional trends exhibited by the two objectives with respect to changes in the number of clusters.

More importantly, even for a fixed number of clusters, there are cases where both single-objective versions will fail, but a multiobjective approach will work. In order to demonstrate this, let us consider the scenario shown in Figure 3. The data set consists of three clusters that contain the same number of data points and are easily discernible to the human eye. The difficulty of the data set arises from two facts: (1) the clusters are constituted from data points with very different densities, and (2) a couple of noise points at the border of the sparse cluster 'bridge the gap' to the smaller two clusters. As a direct result of these properties the assumptions made by overall deviation and connectivity are both violated:

– The correct three-cluster solution does not correspond to the minimum in overall deviation, as splitting of the large cluster results in a much greater reduction in overall deviation.
– The correct three-cluster solution does not correspond to the minimum in connectivity, as the separation of the large from the small clusters involves an increase in connectivity, and the assignment of outliers to their own cluster therefore becomes preferable.

Consequently, the optimization of any *one* of the two objectives, will not lead to the correct solution. Indeed, none of our contestant single-objective algorithms can solve this data set: they all fail to separate the two smaller clusters — instead, *k*-means splits the large cluster in half, and single-link and average-link separate outliers from this cluster.

MOCK in contrast, which explores the *trade-offs* between overall deviation and connectivity, easily manages to detect the correct solution. In particular, in many cases, MOCK can even *discard* those solutions that are optimal only under *overall deviation* or *connectivity*, as these evaluate particularly badly under the respective second objective, and are therefore (due to the natural biases in the measures) likely to be dominated by clustering solutions with a different number of clusters. Overall, the performance of MOCK seems to demonstrate that the

quality of the trade-off between our two objectives is *in many cases* a much better indicator of clustering quality than the individual objectives themselves.

4.3   Performance Curves Versus Pareto Fronts

In this last section, we aim to visualize the different strategies pursued by single-objective clustering algorithms and our two-objective version. Towards this end, we compare the output of MOCK with the solutions obtained by k-means, average-link and single-link agglomerative clustering, when run for a range of different numbers of clusters $k \in 1, \ldots, 25$. We then evaluate all resulting solutions using overall deviation and connectivity, and visualize, in normalized two-objective space, the solutions obtained (see Figure 4). For MOCK, the resulting curve is simply its Pareto front and therefore monotonic. For the other three algorithms the resulting performance plots may contain dominated points, and may therefore be non-monotonic. This is because each of these algorithms only optimizes one of the two objectives: thus, solutions that are better in one objective are not necessarily worse in the other.

A combined analysis of these plots and the F-Measure values provided in Table 2 confirms that the trade-off between overall deviation and connectivity is a good indicator of quality on the data sets employed. There appears to be a distinct correlation between the quality of solutions and their distance to the 'knee' in MOCK's Pareto front. In particular, the algorithms that perform well on a given data set have at least one solution that comes close to or even dominates MOCK's solutions in the 'knee' region. This is generally the solution that best captures the structure of the data set, and also scores highest under the F-Measure. Straightforward examples of this are the k-means and average-link solutions on the *Square4* and *Sizes5* data sets, and the single-link solution on the *Spiral* data set. Results for the *Long1* data set are slightly more involved. Here, k-means and average-link are clearly both widely off track. The F-Measure value for single-link indicates that, due to noise points, the algorithm also has problems to find the correct clustering solution for $k = 2$. This is confirmed by the fact that, in Figure 4, the $k = 2$ solution for single-link is quite far off the 'knee'. However, the plot also reveals (and F-Measure values confirm) that there *are* single-link solutions closer to the optimum: these are the solutions for $k = 4$ or higher, for which single-link assigns noise points to their own clusters but *also* manages to correctly separate the two main clusters present in the data set.

In addition to the above, our visualization demonstrates some general aspects of the algorithms' clustering behaviour. On several data sets, we can observe single-link's tendency to gradually separate outliers, without any significant decrease in overall deviation (even for high numbers of clusters). Clearly, this property makes the algorithm very sensitive to noise, which is one of the reasons why it is rarely applied in practice. k-means, in contrast, quickly reduces overall deviation, but pays virtually no attention to the underlying local data distribution. This becomes evident in a rapid increase and apparent non-monotonicity (e.g. see *Long1*) in connectivity. Only average-link agglomerative clustering shows a more reasonable overall behaviour: its capabilities to satisfy both objectives seem
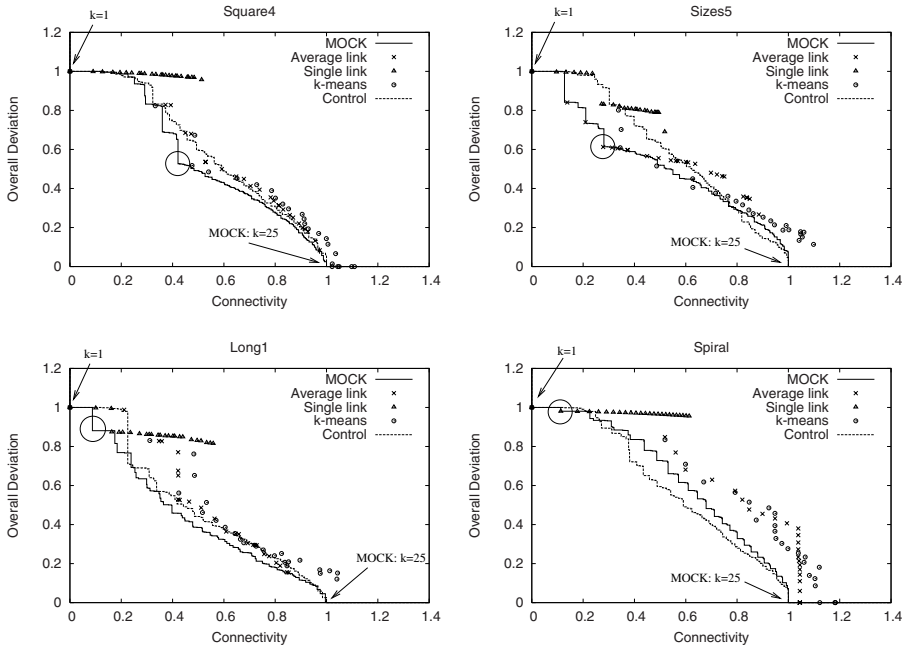
Fig. 4. Plots of all solutions, in normalized objective space, for $k \in [1, \ldots, 25]$ of k-means, single-link and average-link on the *Square4*, *Sizes5*, *Long1* and *Spiral* data sets. Both MOCK's Pareto front and one of its control fronts are visualized as attainment surfaces, and arrows indicate the position of the $k = 1$ solution (identical for all algorithms) and MOCK's $k = 25$ solution. Knees are indicated by circles centred around the solution identified by MOCK

somewhat superior to those of k-means and single-link. For increases in k, it is monotonic in both objectives and its solutions are distributed more uniformly along the Pareto front.

5   Conclusion

Existing clustering algorithms are limited to optimizing (explicitly or otherwise) one single clustering objective. This can lead to a lack of robustness with respect to different data properties, a limitation which we have suggested can be overcome by the use of several complementary objectives. In this paper we have introduced an advanced multiobjective clustering algorithm, MOCK. A comparative study has shown the robustness of the approach both at finding high quality solutions and determining the number of clusters. The origins of MOCK's good performance have then been investigated further. Using an archetypal problem we have demonstrated that the identification of trade-off solutions in clustering can be crucial: the simultaneous optimization of two complementary objectives may permit the solution of clustering problems that are not solvable by either

of the two objectives individually. The differences between single-objective and multiobjective clustering have been further underlined by a visual comparison of the quality of the trade-off solutions generated by single- and multiobjective algorithms for a range of different numbers of clusters. The software for MOCK is available on request from the first author.

References

1. Supporting material for MOCK. http://dbk.ch.umist.ac.uk/handl/mock/
2. J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 722–731. Springer-Verlag, 2004.
3. D. W. Corne, J. D. Knowles, and M. J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kaufmann, 2001.
4. E. Falkenauer. *Genetic Algorithms and Grouping Problems.* John Wiley & Son Ltd, 1998.
5. G. Fleurya, A. Hero, S. Zareparsi, and A. Swaroop. Gene discovery using Pareto depth sampling distributions. *Special Number on Genomics, Signal Processing and Statistics, Journal of the Franklin Institute*, 341(1–2):55–75, 2004.
6. J. Handl and J. Knowles. Evolutionary multiobjective clustering. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pages 1081–1091. Springer-Verlag, 2004.
7. J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. Technical Report COMPYSYBIO-TR-2004-02, Department of Chemistry, UMIST, UK, August 2004.
8. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
9. Y. Kim, W. N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.
10. J. Kleinberg. An impossibility theorem for clustering. In *Proceedings of the 15th Conference on Neural Information Processing Systems*, 2002. http://www.cs.cornell.edu/home/kleinber/nips15.ps.
11. M. H.C Law. Multiobjective data clustering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 424–430. IEEE Press, 2004.
12. U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33:1455–1465, 2000.
13. H. Pan, J. Zhu, and D. Han. Genetic algorithms applied to multi-class clustering for gene expression data. *Genomics, Proteomics & Bioinformatics*, 1(4), 2003.
14. Y.-J. Park and M.-S. Song. A genetic algorithm for clustering problems. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 568–575. Morgan Kaufmann, 1998.

15. J. M. Pena, J. A. Lozana, and P. Larranaga. An empirical comparison of four initialization methods for the k -means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

16. A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, 2002.

17. R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the Gap statistic. Technical Report 208, Department of Statistics, Stanford University, USA, 2000.

18. A. Topchy, A. K. Jain, and W. Punch. Clustering ensembles: Models of consensus and weak partitions. Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

19. C. van Rijsbergen. *Information Retrieval, 2nd edition.* Butterworths, 1979.